The Difference Between Failure and Success is +1

Ioan Raicu
Computer Science Department, Illinois Institute of Technology
Math and Computer Science Division, Argonne National Laboratory

September 3rd, 2015
1st Workshop on E-science ReseaRch leading tO negative Results (ERROR) 2015

DataSys: Data-Intensive Distributed Systems Laboratory

- **Research Focus**
  - Emphasize designing, implementing, and evaluating systems, protocols, and middleware with the goal of supporting **data-intensive applications on extreme scale distributed systems**, from many-core systems, clusters, grids, clouds, and supercomputers
- **People (15)**
  - 1 Faculty member
  - 4 PhD students
  - 4 MS students
  - 4 UG student
  - 2 HS students

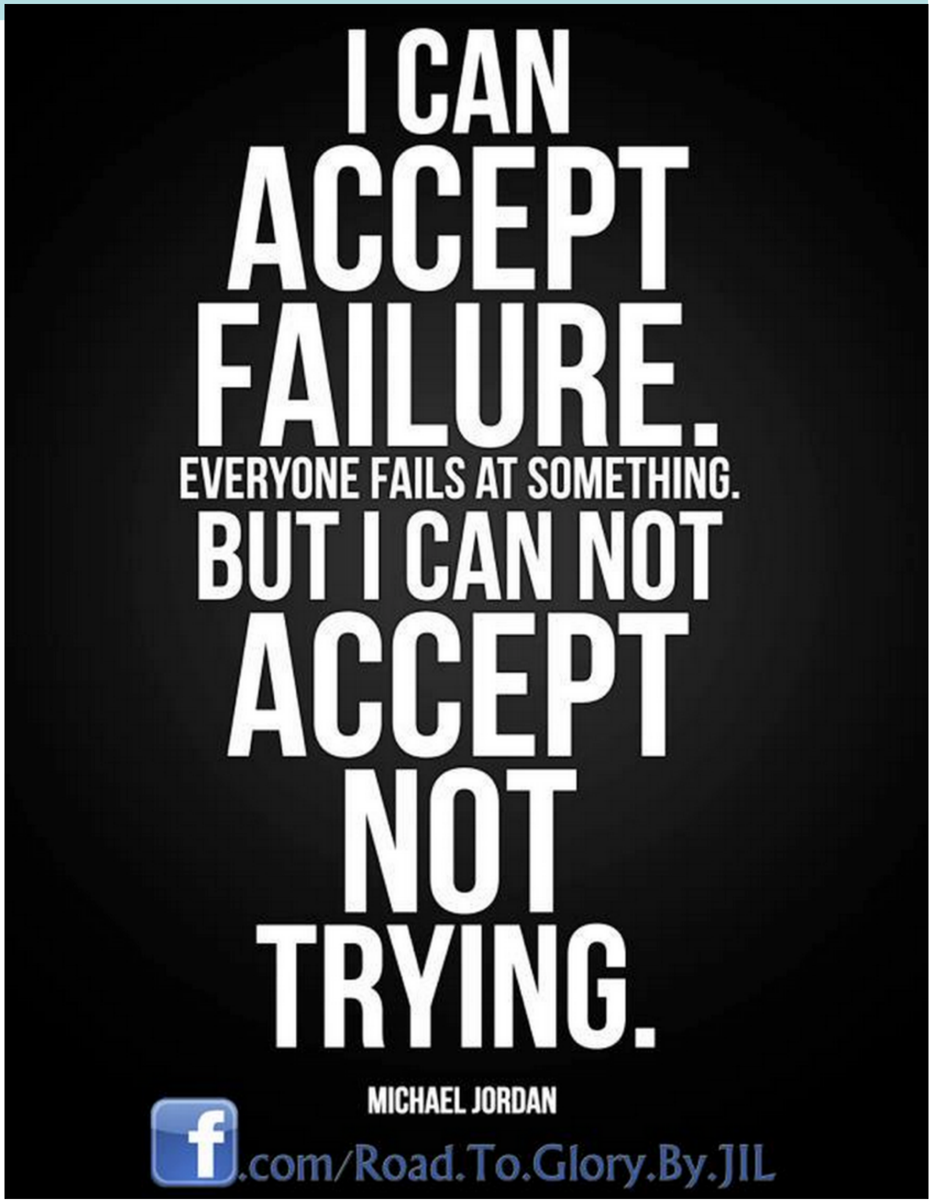
- **Contact**
  - http://datasys.cs.iit.edu/
  - iraicu@cs.iit.edu

The Difference Between Failure and Success is +1

Tenure-Track and Teaching Faculty Positions in CS at Illinois Institute of Technology

Failure is Common

"He, who falls nine times and gets up ten times will succeed"



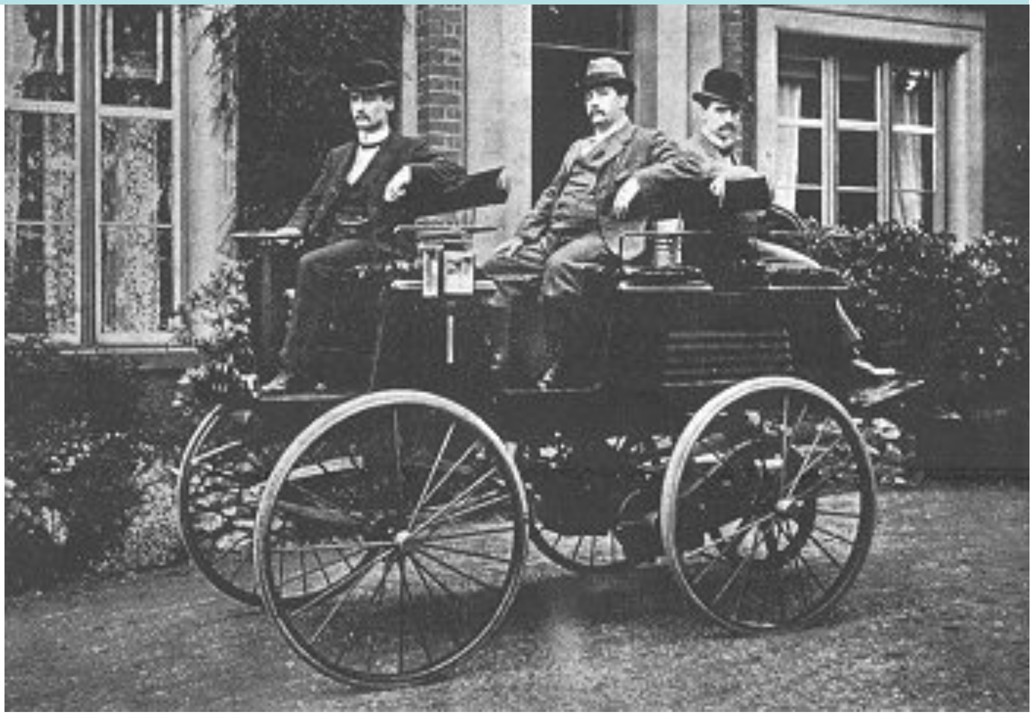
I CAN ACCEPT FAILURE. EVERYONE FAILS AT SOMETHING. BUT I CAN NOT ACCEPT NOT TRYING.

MICHAEL JORDAN

f.com/Road.To.Glory.By.JIL

Many Examples Electric Cars

- First practical electric car built by Thomas Parker [1884]



- German electric car [1904]

Bundesarchiv, Bild 183-1990-1126-500
Foto: o.Ang. | 1904

Many Examples
Electric Cars

- 1900:
  - 40% Steam Power
  - 38% Electric
  - 22% Gasoline
- 1914
  - 99% Gasoline
  - 1% Electric (by 1920, it was 0%)
- 2015
  - 99% Internal combustion
  - 1% Electric (but rising fast)

Many Examples
Electric Cars

- 100+ years later with very little progress
- Tesla Model S [2012]

Tesla's Model S P85D Just Broke Consumer Reports' Ratings System, Scoring 103 Out Of 100 [August 2015]

Many examples Tablet Computers

- Apple Newton [1993]

- Windows Tablet PC [2001]

Many Examples
Tablet Computers

Apple iPad [2010]

What is Research?

- Research comprises "*creative work undertaken on a systematic basis in order to increase the stock of knowledge,… and the use of this stock of knowledge to devise new applications*." It is used to establish or confirm facts, reaffirm the results of previous work, solve new or existing problems, support theorems, or develop new theories. [Wikipedia]

What does Research mean to me?

```
research(problem)
{
    while (ERROR)
    {
        Learn what others have done, including yourself
        Brainstorm on things others/yourself could have done better
        Devise solution for improvements (theory, sim., or systems)
        Implement solution
        Evaluate solution
        Compare against state of the art
    }
    Publish positive results
}
```

What does Research mean to me?

- It means that research is done over the course of many iterations, and often takes many years
- Often times only positive results are published (after likely many failures along the way)
- It is these failures that make us better researchers, as they teach us important lessons
- Everyone fails all the time, but the difference between the winners and the losers is how quickly they get back up, learn from their failure, and try again!

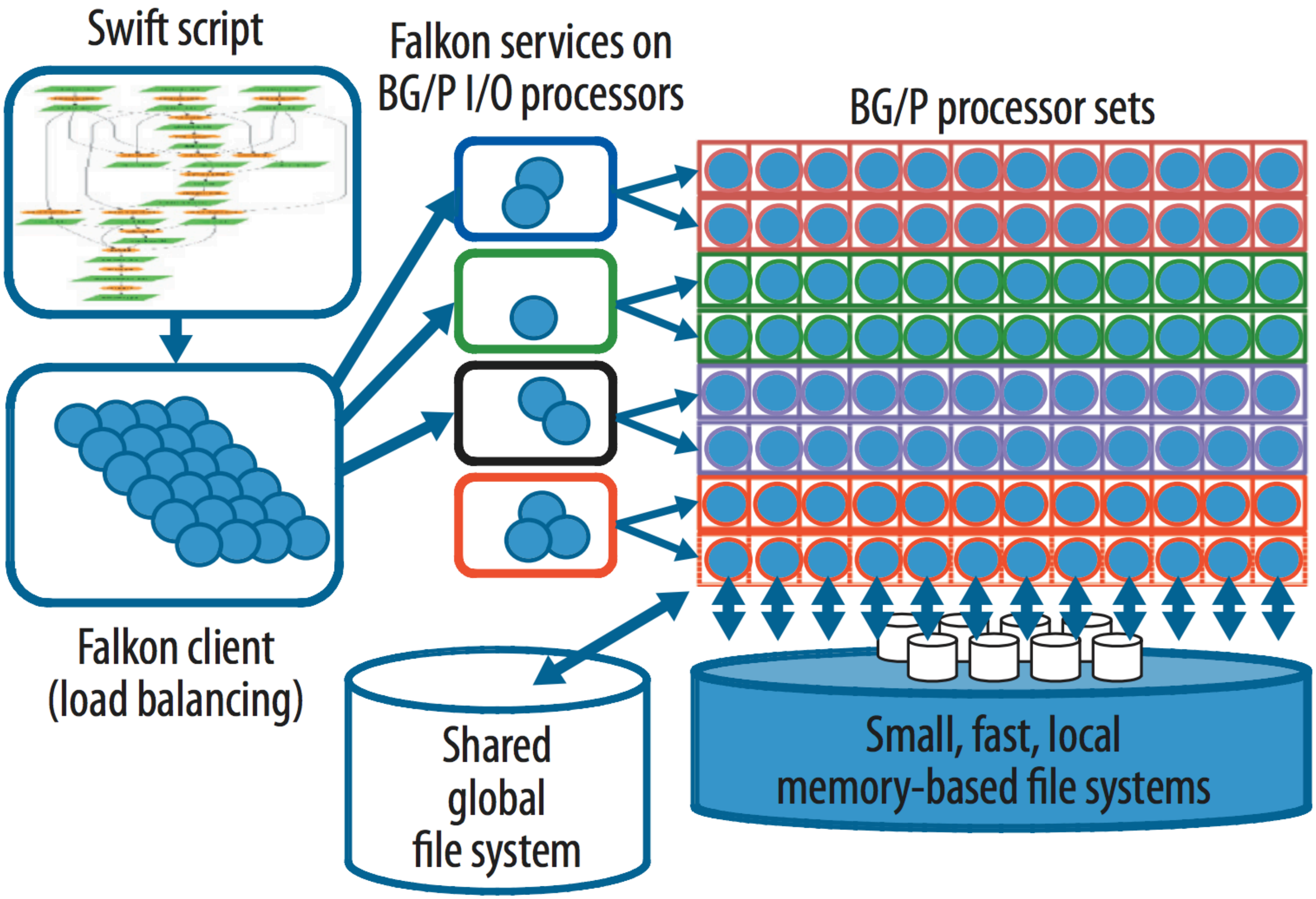
Research Methodology

- **Identify** some PROBLEM from the scientific computing community
  - Most often ➔ **build a system** to solve the PROBLEM
  - Increasingly often ➔ use **simulations** to study complex systems that could solve the PROBLEM, aiming to learn insights into how to build the real system later
  - Sometimes ➔ use **theory** to prove certain properties of the systems we have implemented, or that we want to implement
  - Apply techniques across a range of distributed systems, from **many-core platforms, to clusters, grids, supercomputers, and clouds**
  - Many **collaborators** who work in language design, computer architecture, networking, theory, and domain scientists

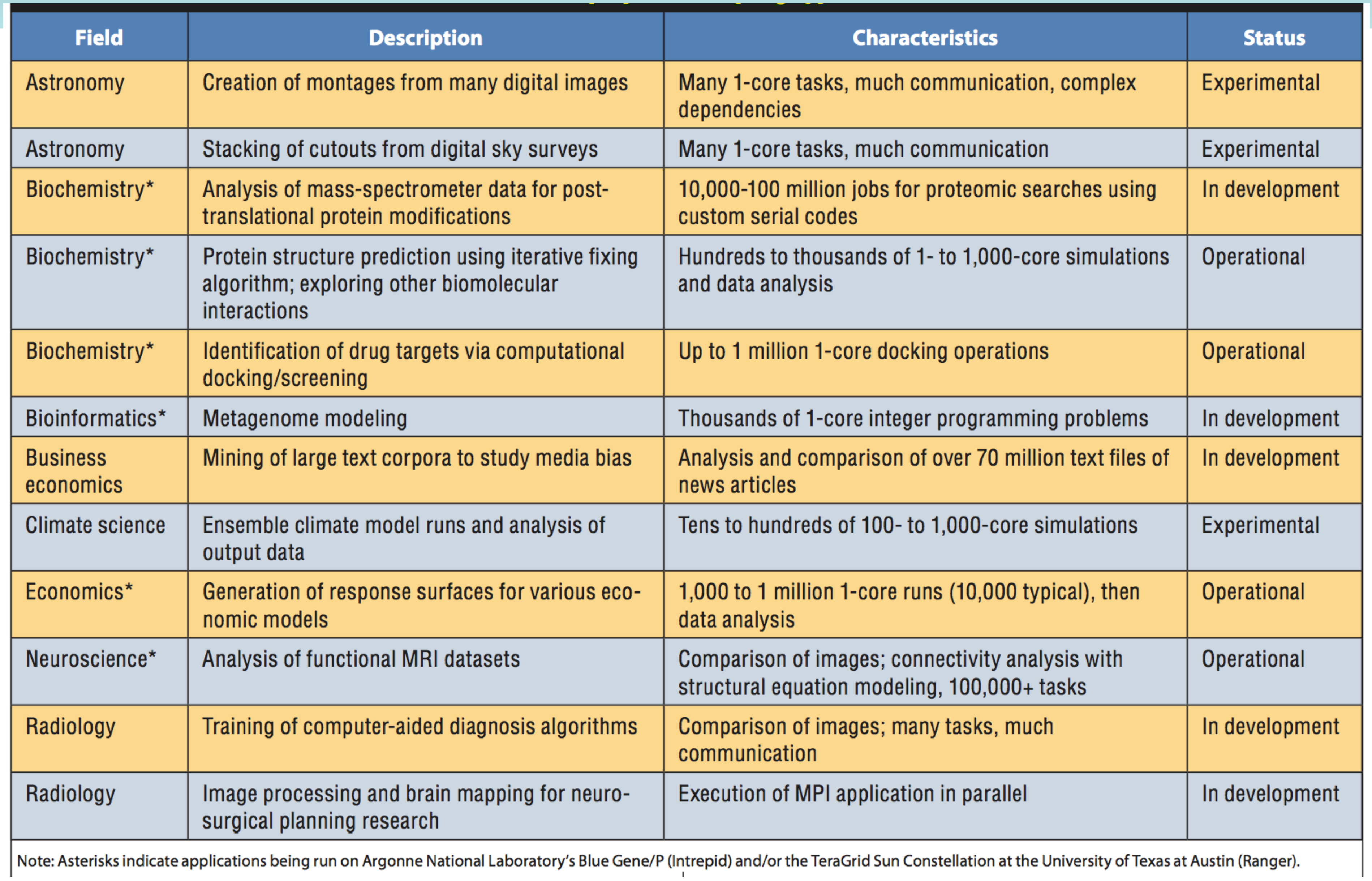
What am I well known for? Many-Task Computing

- Many-Task Computing (MTC)
  - Bridge between high-throughput computing (HTC) and high-performance computing (HPC)
- Examples of MTC systems are parallel programming systems
  - Swift parallel scripting language
  - Others: Hadoop, Spark, Charm++, Legion, Pegasus, Taverna, Nimrod, and many others

Swift & Falkon on a Supercomputer [2007+]


Swift script
Falkon services on BG/P I/O processors
BG/P processor sets
Falkon client (load balancing)
Shared global file system
Small, fast, local memory-based file systems

Scientific Computing Applications



Field	Description	Characteristics	Status
Astronomy	Creation of montages from many digital images	Many 1-core tasks, much communication, complex dependencies	Experimental
Astronomy	Stacking of cutouts from digital sky surveys	Many 1-core tasks, much communication	Experimental
Biochemistry*	Analysis of mass-spectrometer data for post-translational protein modifications	10,000-100 million jobs for proteomic searches using custom serial codes	In development
Biochemistry*	Protein structure prediction using iterative fixing algorithm; exploring other biomolecular interactions	Hundreds to thousands of 1- to 1,000-core simulations and data analysis	Operational
Biochemistry*	Identification of drug targets via computational docking/screening	Up to 1 million 1-core docking operations	Operational
Bioinformatics*	Metagenome modeling	Thousands of 1-core integer programming problems	In development
Business economics	Mining of large text corpora to study media bias	Analysis and comparison of over 70 million text files of news articles	In development
Climate science	Ensemble climate model runs and analysis of output data	Tens to hundreds of 100- to 1,000-core simulations	Experimental
Economics*	Generation of response surfaces for various economic models	1,000 to 1 million 1-core runs (10,000 typical), then data analysis	Operational
Neuroscience*	Analysis of functional MRI datasets	Comparison of images; connectivity analysis with structural equation modeling, 100,000+ tasks	Operational
Radiology	Training of computer-aided diagnosis algorithms	Comparison of images; many tasks, much communication	In development
Radiology	Image processing and brain mapping for neuro-surgical planning research	Execution of MPI application in parallel	In development

Note: Asterisks indicate applications being run on Argonne National Laboratory's Blue Gene/P (Intrepid) and/or the TeraGrid Sun Constellation at the University of Texas at Austin (Ranger).

A Decade of Research in Scheduling 2007 - 2015

- Centralized light-weight MTC scheduling: Falkon [**2007**]
- Centralized data-aware MTC scheduling: Falkon with Data-Diffusion [**2008**]
- Distributed MTC scheduling simulations: SimMatrix [**2013**]
- Distributed MTC scheduling: MATRIX, CloudKon [**2013**]
- MTC scheduling on accelerators: GeMTC [**2014**]
- Distributed data-aware MTC scheduling: MATRIX v2.0 [**2014**]
- Distributed HPC scheduling simulations: SimSlurm++ [**2014**]
- Distributed HPC scheduling: CloudKon, Slurm++ [**2014**]
- Distributed data-aware HPC scheduling: [***2015**]

A Decade of Research in Scheduling 2007

- Swift could easily generate big workflows
  - Parallel for loop that iterated over a dataset with 1000 files; but the 1000 could be millions, or even billions
  - Each iteration could be a separate job or task
  - Swift would interact with grids and supercomputers through standard job submission (e.g. GRAM, Condor)
- In 2007, batch scheduled systems could do O(1) job per second
  - This limited the granularity of the task/jobs in the workflow, and/or concurrency that could be achieved
  - Job submission costs, and unpredictable queue times would often dominate the workflow execution time

A Decade of Research in Scheduling 2007

- To fix the job scheduling problems of 2007 ➔ Falkon
  - Focused on the critical functionality (e.g. light-weight scheduler to handle many single core/node jobs efficiently)
  - Made lightweight by removing resilience, removed HPC-support, and focusing on memory-resident scheduling
  - Avoided unpredictable job queue times by pilot-job abstractions (schedulers within schedulers)
- Falkon achieved 500+ tasks/sec [SC07]
- Swift made significant progress in increasing scales to 100s of nodes and decreasing task granularity to seconds with good efficiency (1~2 orders of magnitude improvement)

A Decade of Research in Scheduling 2008

- The IBM BlueGene/P Supercomputer comes online; goal is to run Swift on 40K nodes with 160K cores
  - Centralized architecture breaks down O(10K-nodes/cores)
    - Cost (both memory and processing power) of having 10K sockets open and active
    - Cost of scheduling, hence the task granularity suffers
    - Resilience becomes more important
    - Stable operation up to 1K-nodes/cores
  - Change Falkon architecture to be hierarchical
    - Allowed near linear scalability at the expense of higher latency in short tasks
    - Potential for load imbalance due to early task placement commitment (long tailed experiments)
    - Achieved 3000+ tasks/sec at 160K-core scales [SC08]

A Decade of Research in Scheduling 2009

- Once scheduling costs were reduced, storage began being the primary bottleneck
  - Storage was remote, leading to high latency
  - Network is a shared resource, and hence not scalable
  - File system metadata is centralized leading to extremely poor performance for metadata intensive workloads
- In 2009 we extended Falkon with data-diffusion [HPDC09]
  - Investigated leveraging many local disks on nodes
  - Kept a centralized index for data management
  - Extended centralized Falkon scheduler to be data-aware
- This work spawned my interests in storage systems

A Decade of Research in Scheduling Centralized → Distributed

- Needed support for finer granularity tasks (sub-second) at extreme scales (1M-cores/nodes)
  - Needed to support orders of magnitude higher tasks/sec throughput performance
  - Completely decentralized techniques (e.g. work stealing)
  - Load balancing goes from trivial in centralized Falkon to a complex and brittle problem

A Decade of Research in Scheduling 2009 - 2014

- 2009: began work on Falkon v2.0 extending the hierarchical system
- 2010: gave up on Falkon, and started from scratch on a system called MATRIX to study work stealing
- 2012: no concrete publishable results on MATRIX, and I turned to simulations (SimMatrix)
- 2012: Reboot with a clean slate design using ZHT (a distributed key/value store) as a building block
- 2013: Publications on both SimMatrix [HPC13] (up to 1M-nodes) and MATRIX [IIT MS13] (up to 1K-cores)
- 2014: Distributed scheduling using message queues (as opposed to key/value stores) ➔ CloudKon [CCGrid 2014]

A Decade of Research in Scheduling 2014 - 2015

- 2014: HPC workloads with Slurm++ [HPDC14, HPDC15]
- 2014: Distributed Data-aware scheduling [BigData14, CCPE15]
- 2015: Investigated Ha
  MATRIX [Cluster15]
- 2015: PhD defense fo
  - Scalable Resource Ma
    Extreme-Scale Distribu

A Decade of Research in Scheduling 2016 and Beyond

- Prototypes to production systems
- Data-aware HPC scheduling to support burst buffer based HPC architectures
- Cross pollinate between supercomputing and cloud research ➔ Big Data in cloud computing has the potential for some very big impact for many disciplines
- Many companies in industry are jumping on board
  - Microsoft announced the acquisition of Mesosphere $1B [August 2015]
  - IBM announced the commitment of 3500 researchers and developers to Spark

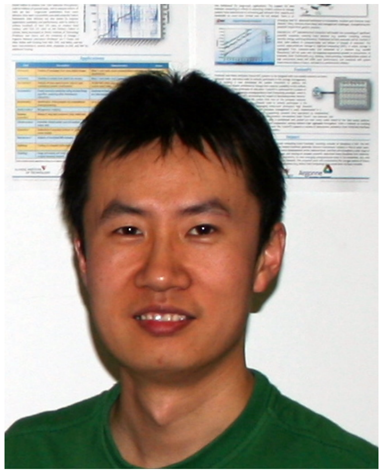
A Decade of Research in Scheduling 2016 and Beyond

- Commercialization
  - Hadoop valued at $1B in 2012, estimated to be $20B in 2018
  - Microsoft announced the acquisition of Mesosphere $1B [August 2015]
  - IBM announced the commitment of 3500 researchers and developers to Spark based projects/products [2015]

A Decade of Research in Storage 2008 - 2015

- Centralized object store: Data-Diffusion in Falkon [**2008**]
- Hybrid file systems: HyCache [**2013**]
- Distributed key/value store: ZHT [**2013**]
- Distributed file system: FusionFS [**2014**]
  - Distributed metadata management [**2014**]
  - Information dispersal algorithms: IStore [**2013**]
  - Provenance: FusionProv [**2013**]
  - Cooperative caching: HyCache+ [**2014**]
  - Dynamic compression: Virtual Chunks [**2014**]
  - Distributed Indexing [*__2015__]
  - Simulations: FusionSim [**2015**]
- Distributed graph database: Graph/Z [**2015**]
- Distributed relational databases: HRDBMS [**2015**]
- Distributed message queues: Fabriq [*__2015__]
- Multi-path network protocol simulations [*__2015__]

Dongfang Zhao
PhD 2015


Tonglin Li
PhD 2015*

A Decade of Research in Storage 2016 and Beyond

- Commercialization
  - Redhat acquires Inktank (the company behind Ceph) for $175M in 2014
  - Intel invests $38M in Databricks (used to accelerated Spark) [2015]
  - Cleversafe develops novel erasure coded large-scale storage systems over the course of a decade ($100M in venture capital so far) [2004 – 2015]

Difference between Failure and Success?

- Persistence to try one more time (+1)
- Learn from mistakes
- Collaborate to gain knowledge faster
- Surround yourself with smart people
- Narrow scope of problem/solution until successful; then broaden scope
- Use trends to help predict potential problems and/or solutions
- Be passionate about your work

More Information

- More information:
  - http://www.cs.iit.edu/~iraicu/
  - http://datasys.cs.iit.edu/
- Contact:
  - iraicu@cs.iit.edu
- Questions?